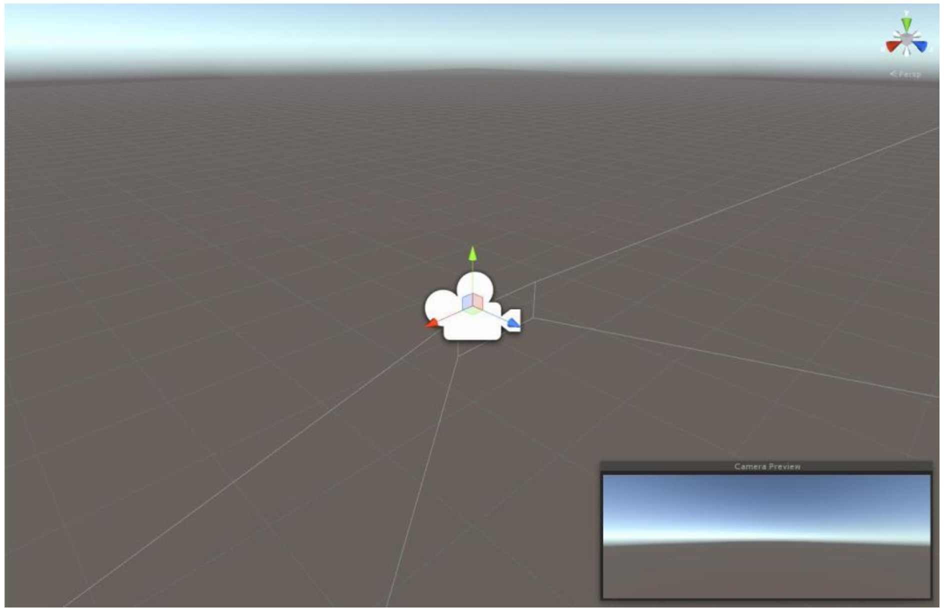
Viewing

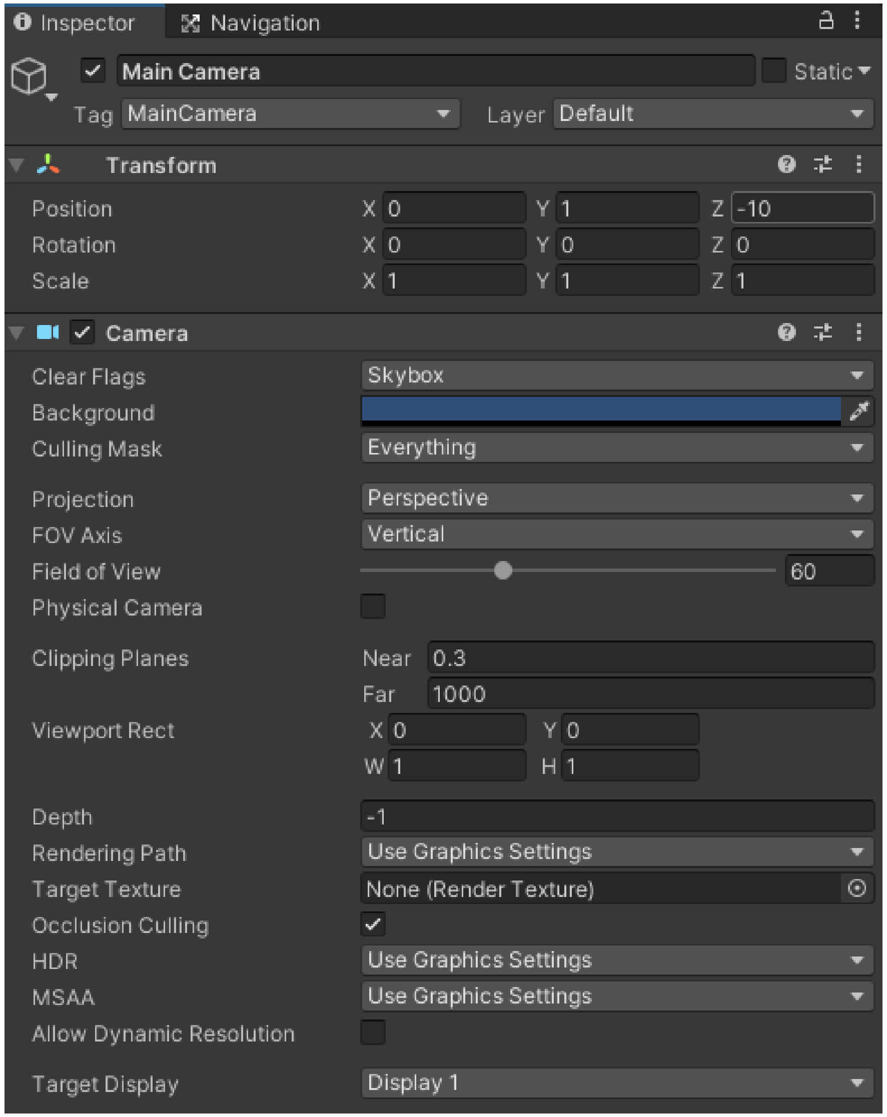
Fall 2025
11/6/2025
Kyoung Shin Park
Computer Engineering
Dankook University

Camera in Unity

- Camera
  - A Unity scene represents GameObjects in a three-dimensional space. Since the viewer's screen is two-dimensional, Unity needs to **capture a view** and "flatten" it for display. It does this using **cameras**.
  - In Unity, you create a camera by adding a Camera component to a GameObject.



Camera Components



Inspector | Navigation

Main Camera | Static

Tag MainCamera | Layer Default

Transform

Position	X 0	Y 1	Z -10
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Camera ✓

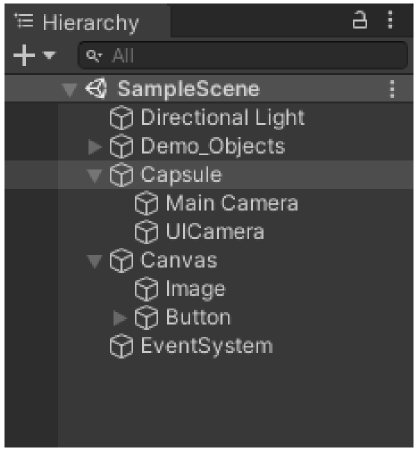
Clear Flags	Skybox
Background	🟦
Culling Mask	Everything
Projection	Perspective
FOV Axis	Vertical
Field of View	60
Physical Camera	☐
Clipping Planes	Near 0.3
	Far 1000
Viewport Rect	X 0 Y 0
	W 1 H 1
Depth	-1
Rendering Path	Use Graphics Settings
Target Texture	None (Render Texture)
Occlusion Culling	✓
HDR	Use Graphics Settings
MSAA	Use Graphics Settings
Allow Dynamic Resolution	☐
Target Display	Display 1

Camera Components

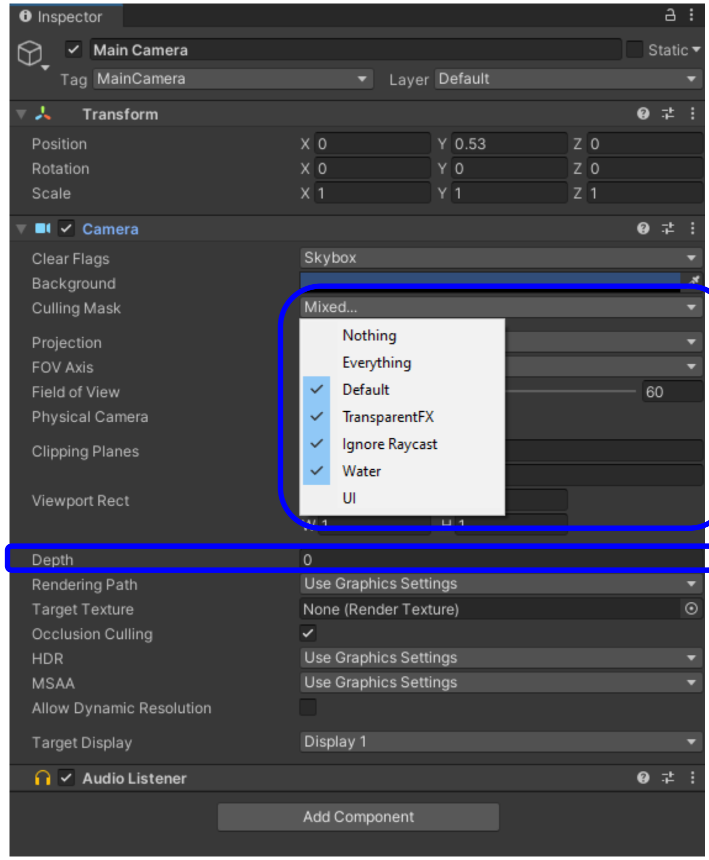
Property:	Function:
Clear Flags	Determines which parts of the screen will be cleared. This is handy when using multiple Cameras to draw different game elements. Skybox is the default setting.
Background	The color applied to the remaining screen after all elements in view have been d rawn and there is no **skybox**.
Culling Mask	Includes or omits layers of objects to be rendered by the Camera. Assigns layers to your objects in the Inspector.
Projection	Toggles the camera's capability to simulate perspective.
Perspective	Camera will render objects with perspective intact.
Orthographic	Camera will render objects uniformly, with no sense of perspective. **NOTE:** Deferr ed rendering is not supported in Orthographic mode. **Forward rendering** is alw ays used.
Size (when Orthographic i s selected)	The **viewport** size of the Camera when set to Orthographic.
FOV Axis (when Perspectiv e is selected)	Field of view axis.
Horizontal	The Camera uses a horizontal field of view axis.
Vertical	The Camera uses a vertical field of view axis.
Field of view (when Persp ective is selected)	The Camera's view angle, measured in degrees along the axis specified in the **FO V Axis** drop-down.

UI Camera Setting in Unity

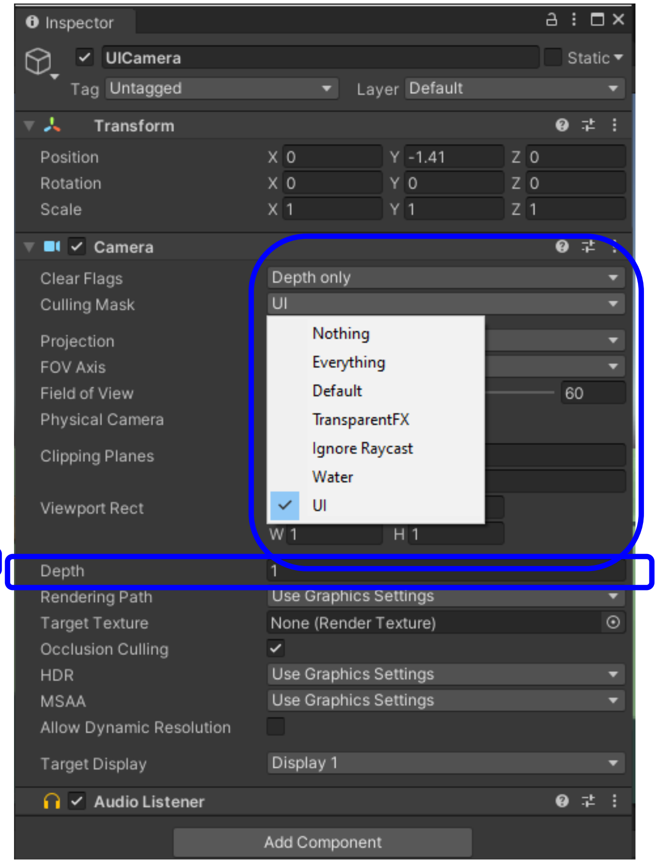
- Capsule(Player) contains **Main Camera** & **UI Camera**
  - Main Camera – *uncheck UI* in **CullMask**, set **Depth** to *0*
  - UI Camera – select *Depth only* in **Clear Flags**, *check only UI* in **Cull Mask**, set **Depth** to *1*



```
Hierarchy
+ ▾  All
▾ SampleScene
    Directional Light
  ▸ Demo_Objects
  ▾ Capsule
      Main Camera
      UICamera
  ▾ Canvas
      Image
    ▸ Button
    EventSystem
```



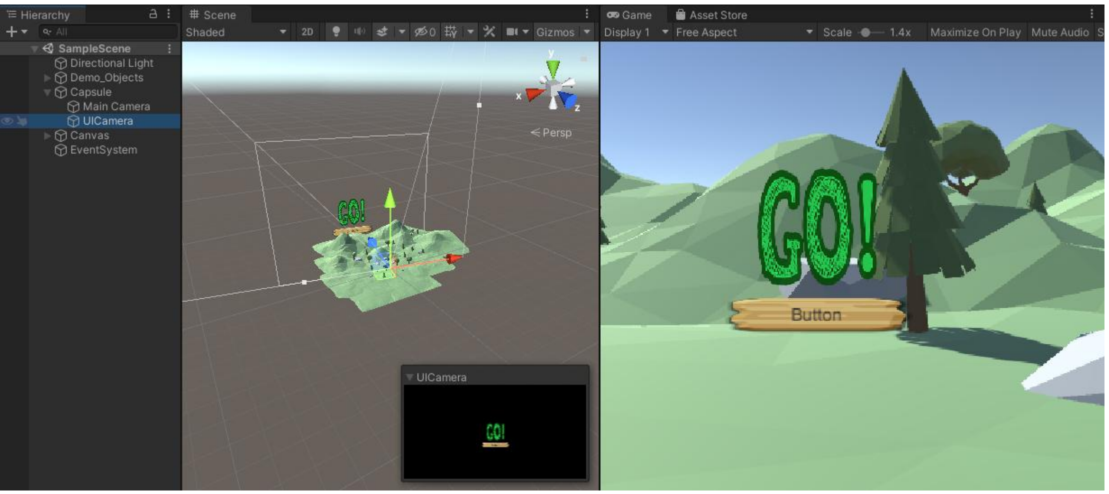
```
Inspector
✓ Main Camera                          Static
Tag MainCamera            Layer Default
Transform
Position          X 0     Y 0.53   Z 0
Rotation          X 0     Y 0      Z 0
Scale             X 1     Y 1      Z 1
✓ Camera
Clear Flags       Skybox
Background
Culling Mask      Mixed...
                    Nothing
                    Everything
                  ✓ Default
                  ✓ TransparentFX
                  ✓ Ignore Raycast
                  ✓ Water
                    UI
Projection
FOV Axis
Field of View                      60
Physical Camera
Clipping Planes
Viewport Rect     W 1     H 1
Depth             0
Rendering Path    Use Graphics Settings
Target Texture    None (Render Texture)
Occlusion Culling ✓
HDR               Use Graphics Settings
MSAA              Use Graphics Settings
Allow Dynamic Resolution
Target Display    Display 1
✓ Audio Listener
Add Component
```

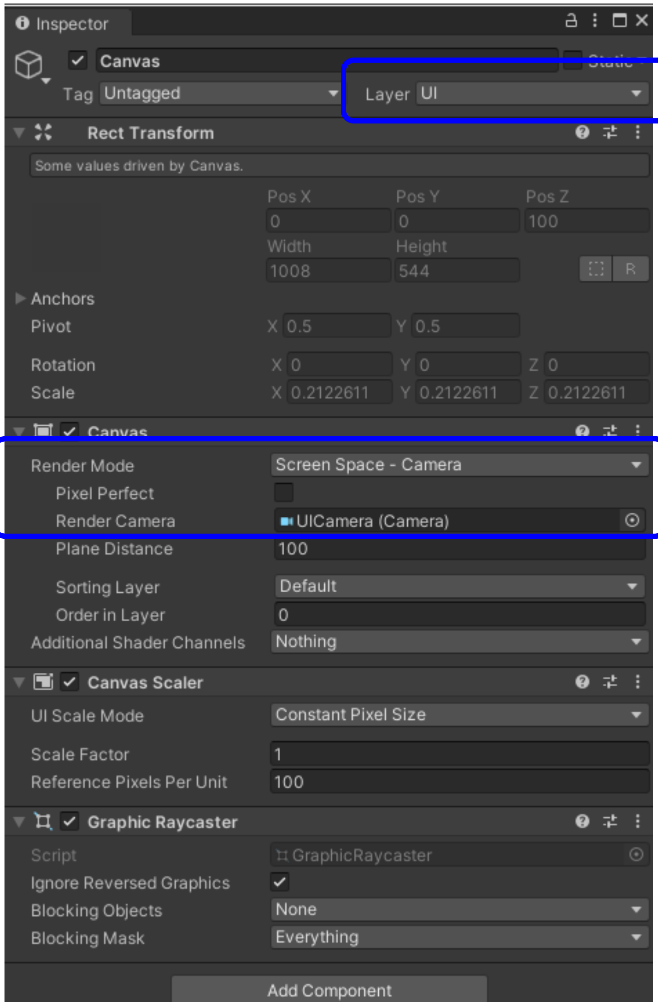


```
Inspector
✓ UICamera                             Static
Tag Untagged              Layer Default
Transform
Position          X 0     Y -1.41  Z 0
Rotation          X 0     Y 0      Z 0
Scale             X 1     Y 1      Z 1
✓ Camera
Clear Flags       Depth only
Culling Mask      UI
                    Nothing
                    Everything
                    Default
                    TransparentFX
                    Ignore Raycast
                    Water
                  ✓ UI
Projection
FOV Axis
Field of View                      60
Physical Camera
Clipping Planes
Viewport Rect     W 1     H 1
Depth             1
Rendering Path    Use Graphics Settings
Target Texture    None (Render Texture)
Occlusion Culling ✓
HDR               Use Graphics Settings
MSAA              Use Graphics Settings
Allow Dynamic Resolution
Target Display    Display 1
✓ Audio Listener
Add Component
```

UI Camera Setting in Unity

- Canvas – set **Layer** to *UI*, set **Render Mode** to *Screen Space – Camera*, set **Render Camera** to *UICamera*
  - Then, Main Camera renders a scene
  - while UI Camera renders UI only.




Inspector

Canvas — Tag Untagged — Layer UI

Rect Transform

Some values driven by Canvas.

	Pos X	Pos Y	Pos Z
	0	0	100
	Width	Height	
	1008	544	
Anchors			
Pivot	X 0.5	Y 0.5	
Rotation	X 0	Y 0	Z 0
Scale	X 0.2122611	Y 0.2122611	Z 0.2122611

Canvas

Render Mode	Screen Space - Camera
Pixel Perfect	
Render Camera	UICamera (Camera)
Plane Distance	100
Sorting Layer	Default
Order in Layer	0
Additional Shader Channels	Nothing

Canvas Scaler

UI Scale Mode	Constant Pixel Size
Scale Factor	1
Reference Pixels Per Unit	100

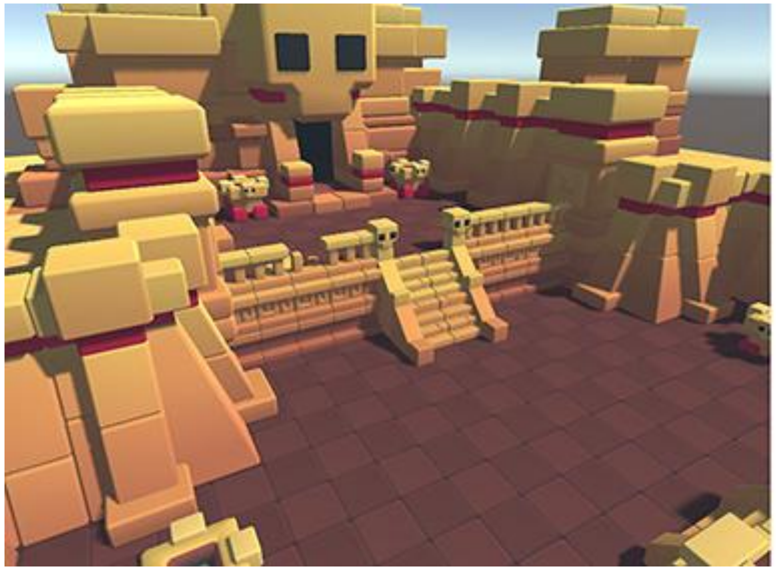
Graphic Raycaster

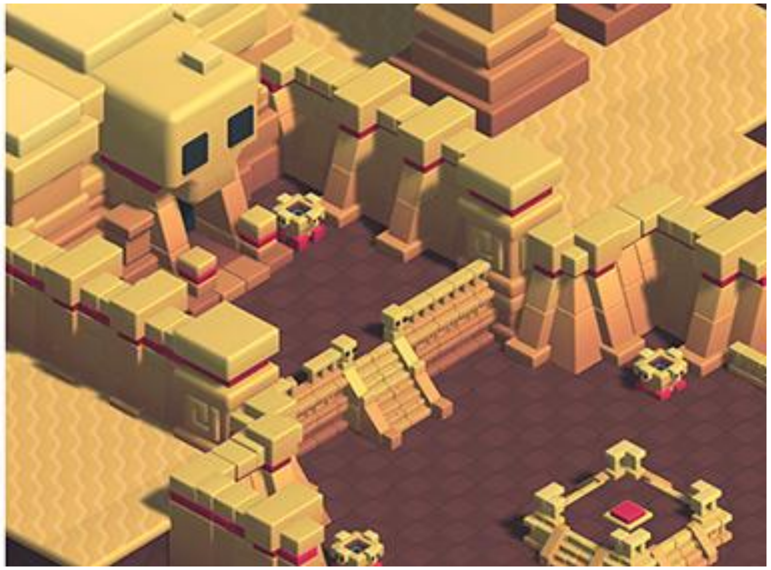
Script	GraphicRaycaster
Ignore Reversed Graphics	✓
Blocking Objects	None
Blocking Mask	Everything

Add Component

Viewer's Perspective

- Before rendering the environment on the screen we consider the camera input such as (**field of view**, **Projection** mode [Orthographic or Perspective]).

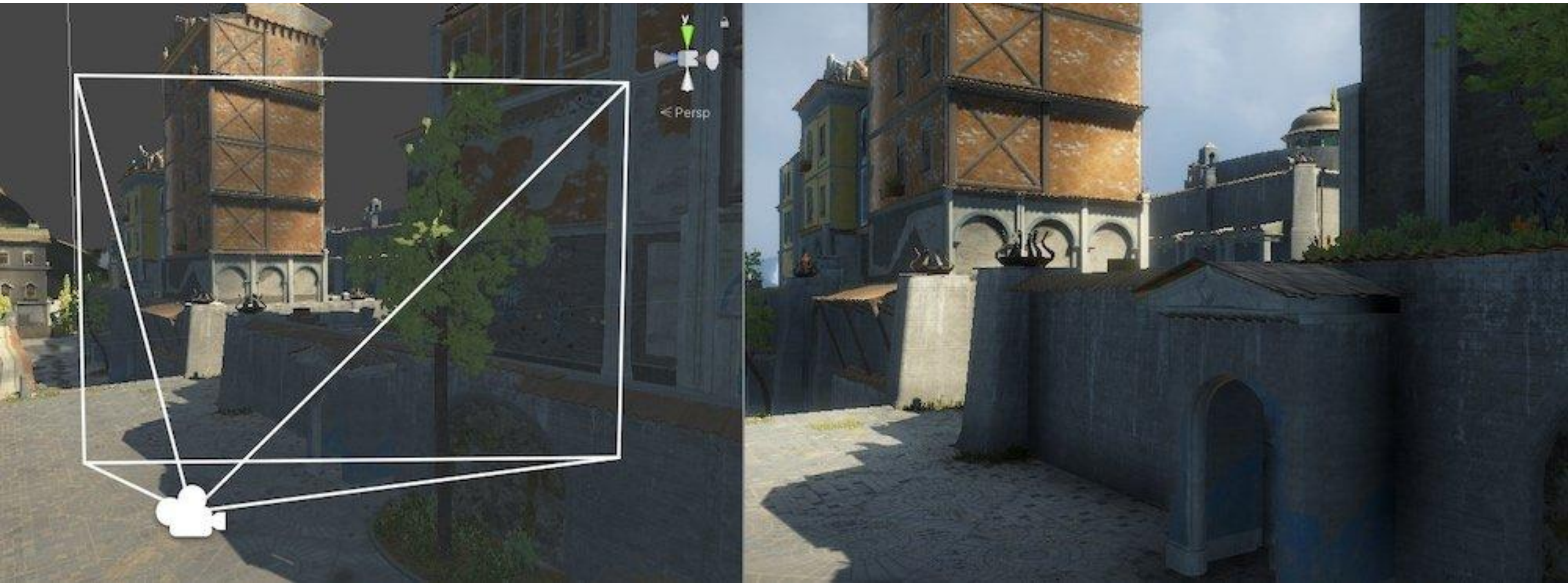

Perspective camera


Orthographic camera

https://docs.unity3d.com/Manual/CamerasOverview.html

Field of View

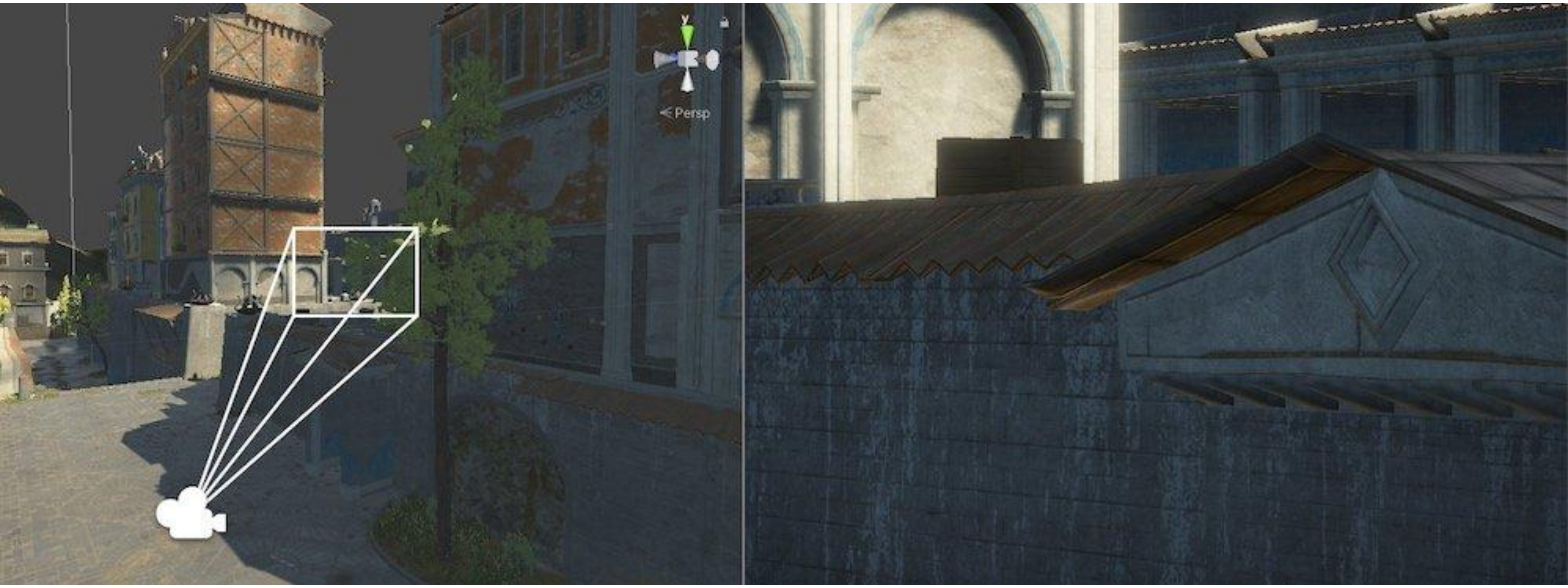
- A wide field of view shows more of the scene.



https://gamedevbeginner.com/how-to-zoom-a-camera-in-unity-3-methods-with-examples/

Field of View

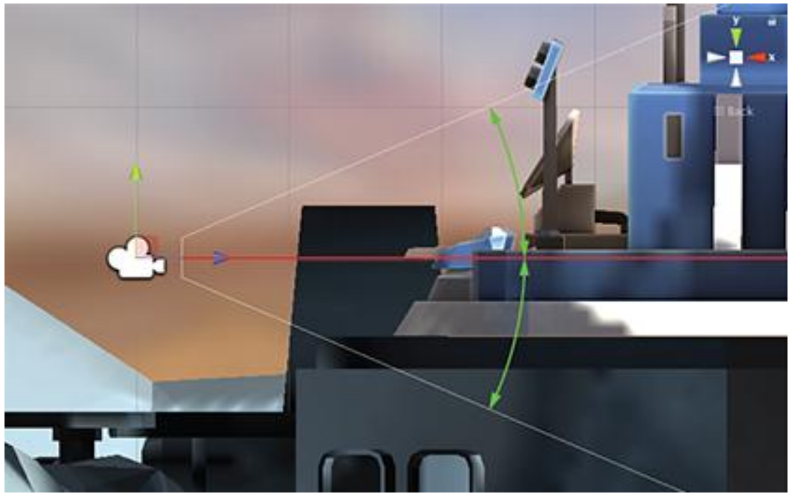
- A narrow field of view shows less of the camera image, zooming it in scene.

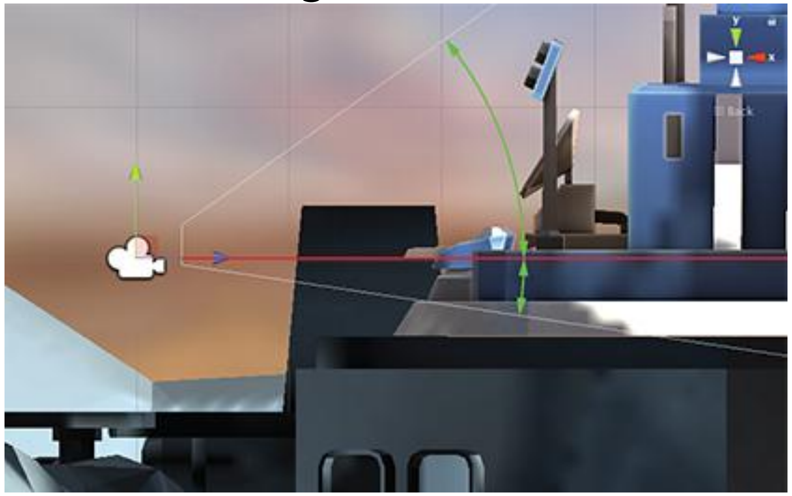


https://gamedevbeginner.com/how-to-zoom-a-camera-in-unity-3-methods-with-examples/

Camera Components

- The camera component's Physical Camera properties simulate real-world camera formats on a Unity camera.
  - Focal Length
    - Smaller focal lengths result in a larger FOV and vice versa
  - Sensor Type & Sensor Size
  - Lens Shift & Gate Fit
    - Camera frustum before (left) and after (right) a Y-axis lens shift



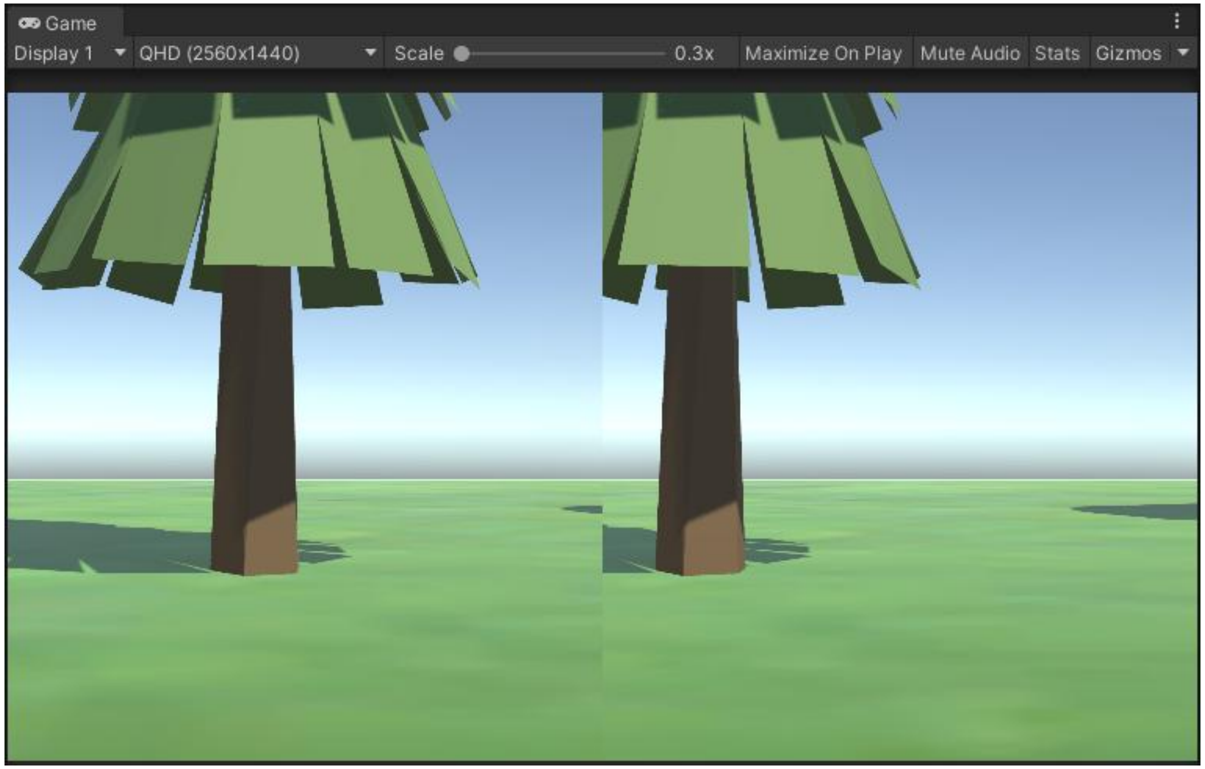


Camera Components

Property:	Function:
Clipping Planes	Distances from the camera to start and stop rendering.
Near	The closest point relative to the camera that drawing will occur.
Far	The furthest point relative to the camera that drawing will occur.
Viewport Rect	Four values that indicate where on the screen this camera view will be drawn. Measured in Viewport Coordinates (values 0–1).
X	The beginning horizontal position that the camera view will be drawn.
Y	The beginning vertical position that the camera view will be drawn.
W (Width)	Width of the camera output on the screen.
H (Height)	Height of the camera output on the screen.
Depth	The camera's position in the draw order. Cameras with a larger value will be d rawn on top of cameras with a smaller value.
Rendering Path	Options for defining what rendering methods will be used by the camera.
Forward	Forward is the traditional rendering path.
Deferred *Lighting*	*Deferred Shading* is the rendering path with the most lighting and shadow fidelity, and is best suited if you have many realtime lights. It requires a certain level of hardware support.
Legacy Vertex Lit	*Legacy Vertex Lit* is the rendering path with the lowest lighting fidelity and no support for realtime shadows. It is a subset of Forward rendering path.
Legacy *Deferred*	*Legacy Deferred (light prepass)* is similar to Deferred Shading, just using a different technique with different trade-offs.

Viewport

- Viewport
  - The space set inside the window. Drawing is restricted to inside the viewport.



Camera1 Viewport Rect
X:0 Y: 0 W:0.5 H:1

Camera2 Viewport Rect
X:0.5 Y: 0 W:0.5 H:1

Camera Components

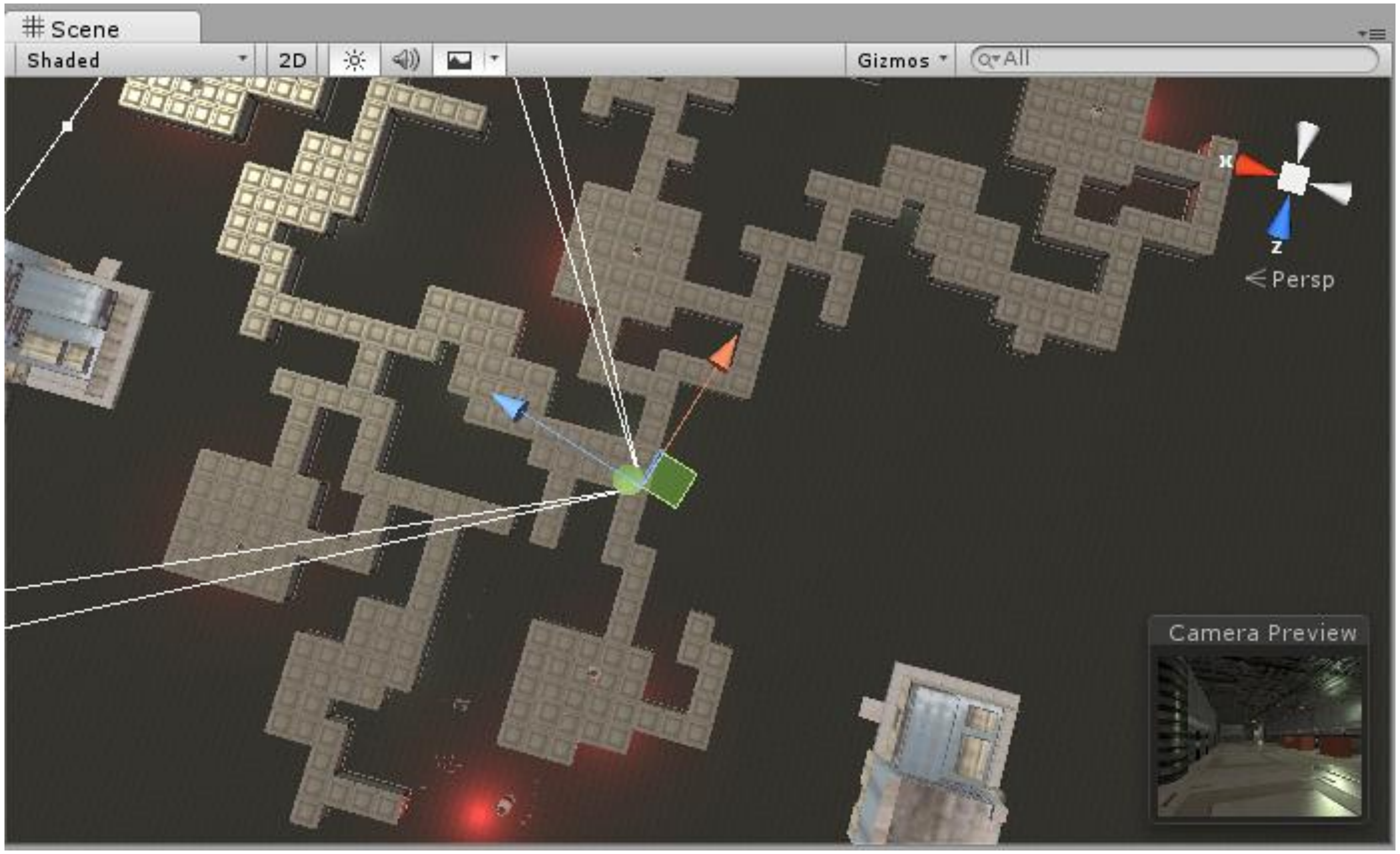
Property:	Function:
Target Texture	Reference to a Render Texture that will contain the output of the Camera view . Setting this reference will disable this Camera's capability to render to the sc reen.
Occlusion Culling	Enables Occlusion Culling for this camera. Occlusion Culling means that objec ts that are hidden behind other objects are not rendered, for example if they are behind walls.
Allow HDR	Enables High Dynamic Range rendering for this camera.
Allow MSAA	Enables multi sample **antialiasing** for this camera.
Allow Dynamic Resoluti on	Enables Dynamic Resolution rendering for this camera.
Target Display	Defines which external device to render to. Between 1 and 8.

Occlusion Culling

- **Occlusion Culling** is a feature that disables rendering of objects when they are not currently seen by the camera because they are obscured (occluded) by other objects.
- This does not happen automatically in 3D computer graphics since most of the time objects farthest away from the camera are drawn first and closer objects are drawn over the top of them (this is called "overdraw").
- Occlusion Culling is different from Frustum Culling.
- Frustum Culling only disables the renderers for objects that are outside the camera's viewing area but does not disable anything hidden from view by overdraw.
- Note that when you use Occlusion Culling you will still benefit from Frustum Culling.

Occlusion Culling

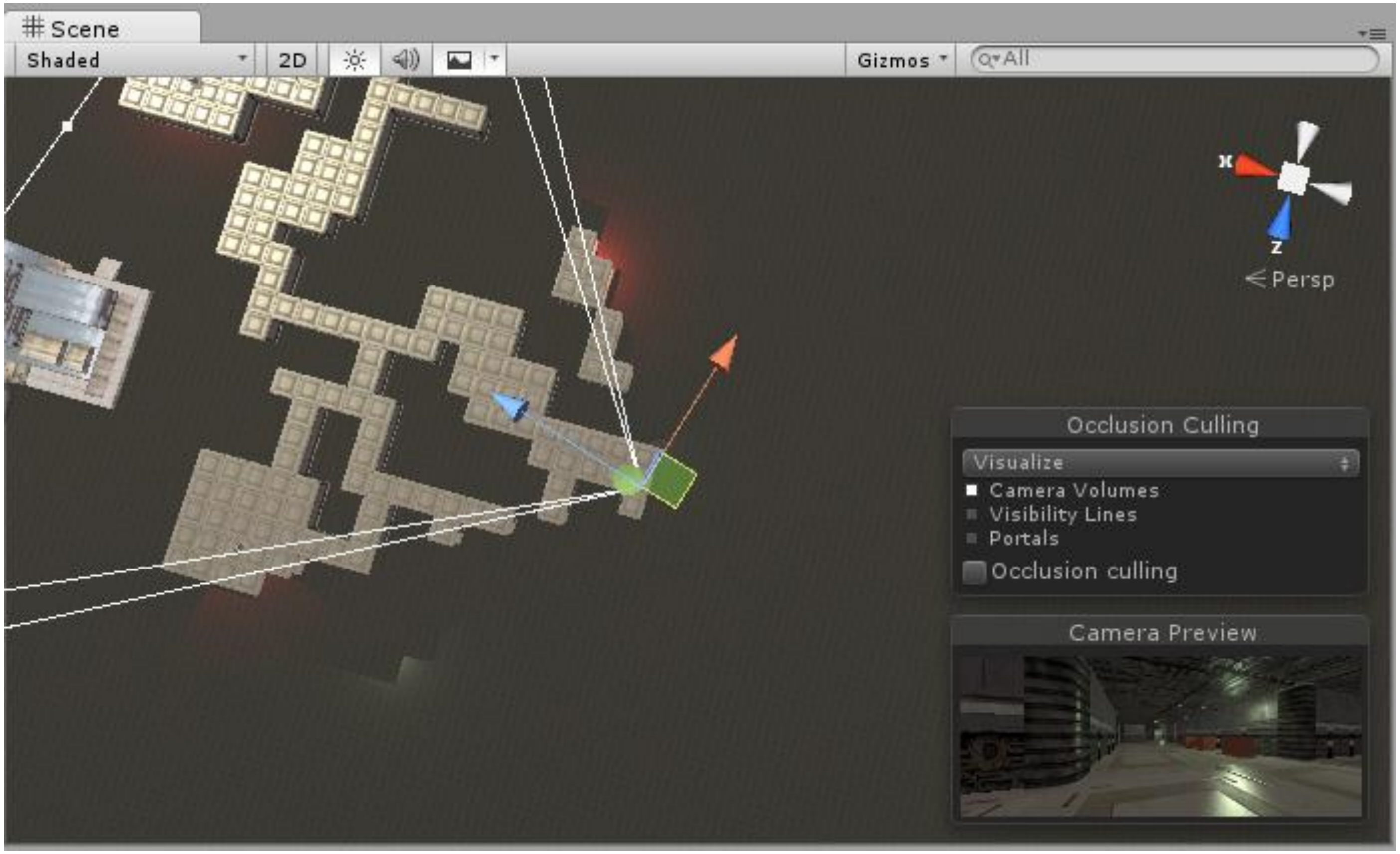
- A maze-like indoor level. This normal scene view **shows all visible Game Objects**.



https://docs.unity.com/Manual/OcclusionCulling.html

Occlusion Culling

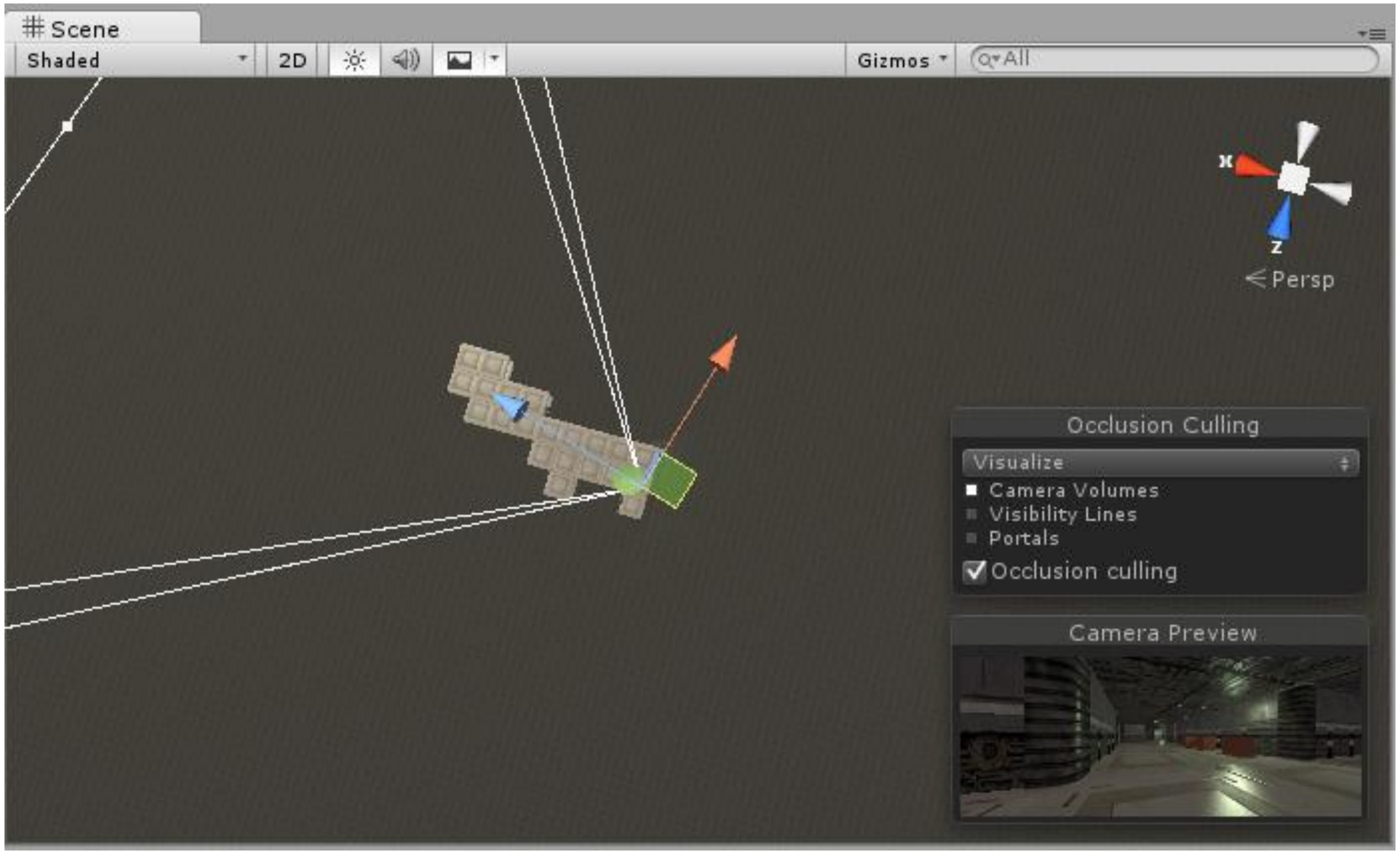
- Regular **frustum culling** renders **all Renderers within the Camera’s view**.



https://docs.unity.com/Manual/OcclusionCulling.html

Occlusion Culling

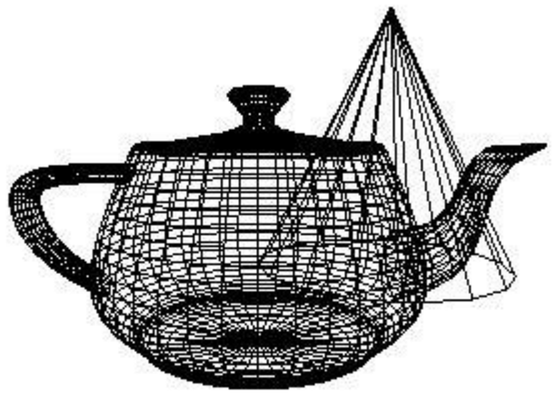
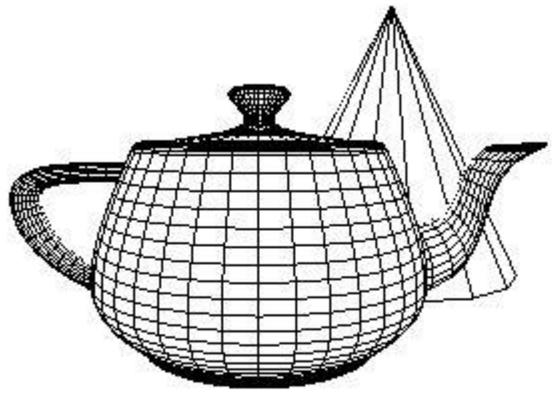
- **Occlusion culling** removes Renderers that are entirely obscured by nearer Renderer.



https://docs.unity.com/Manual/OcclusionCulling.html

Hidden Surface

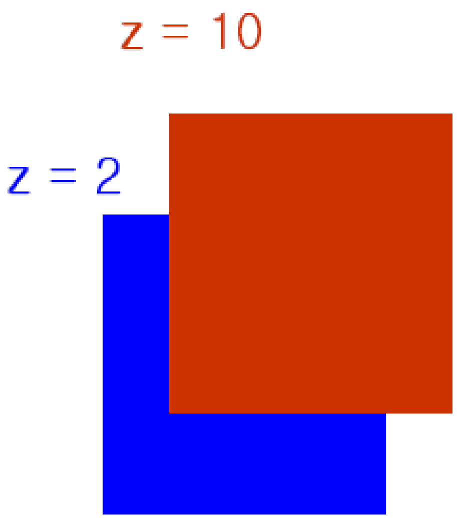
- Hidden surfaces provides the occlusion depth cue.
- In computer graphics, the term occlusion refers to objects that are close to the viewer to occlude objects that are far from the viewer.
- In the graphics pipeline, hidden surface removal is performed before shading and rasterization with occlusion culling.

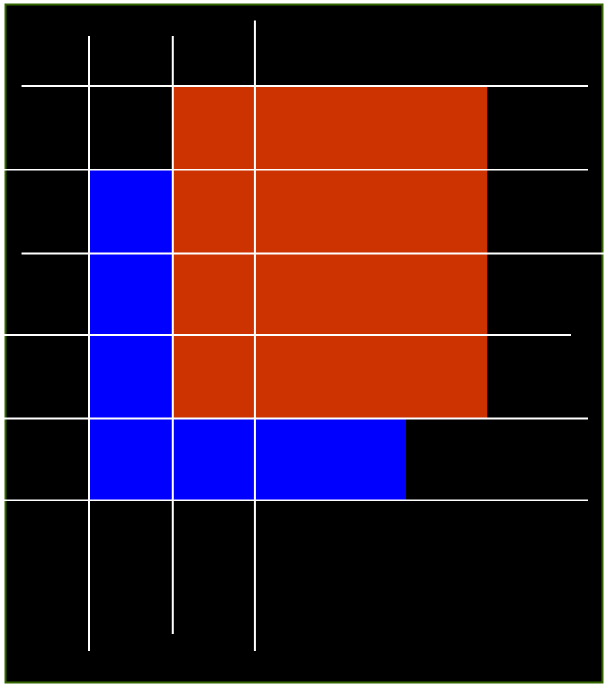



Hidden Surface Removal

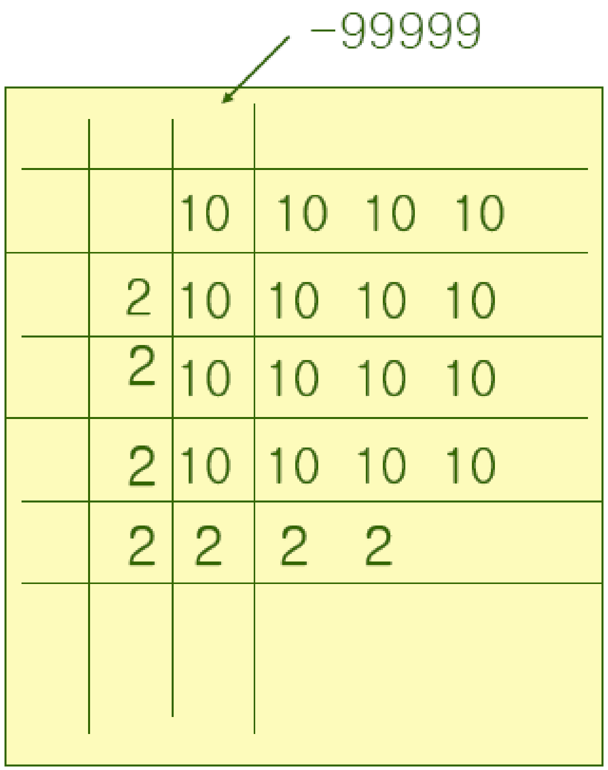
- Hidden Surface Removal Algorithm
  - Object space technique – compare objects or parts of objects to determine which side and line are not visible as a whole.
    - Depth-sorting algorithm – After aligning each side of the polygon according to the depth, it is drawn from the far one to front one. Also known as Painter’s algorithm.
    - Binary Space Partitioning (BSP) tree – Using BSP tree, the space is continuously partitioned by separating front and back according to the viewer direction.
  - Image space technique – act as part of the projection, and visibility is determined in units of points at the location of object pixels on each projection line.
    - Z-buffer (depth buffer) – This is the most commonly used image space technique. By examining the visibility of an object in pixels, it draws the value of the plane with the smallest z (depth) value. We need a depth buffer (z-buffer) to store the z-value.
    - Ray-casting – It projects light (ray) through each pixel on the projection surface at the viewpoint, selects the object that first meets this light and draws the pixel. It is an effective hidden surface removal algorithm for curved surface.

Z-buffer





Color buffer

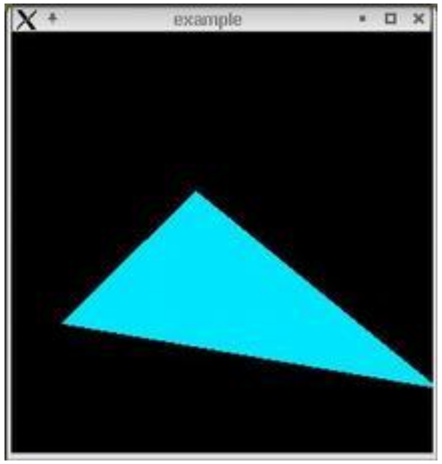


		10	10	10	10
	2	10	10	10	10
	2	10	10	10	10
	2	10	10	10	10
	2	2	2	2	

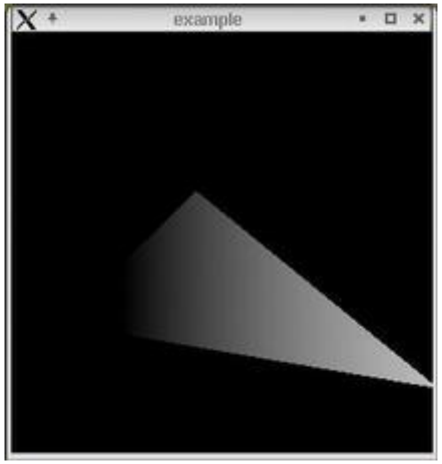
Z-buffer
(depth buffer)

Z-buffer

- Polygon rendering means eventually being filled with pixels.
- The color buffer contains RGB color per pixel to be drawn.
- The depth buffer (Z-buffer) has depth information per pixel to be drawn.



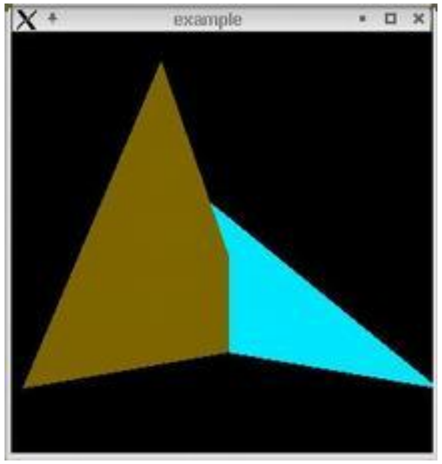
Color buffer



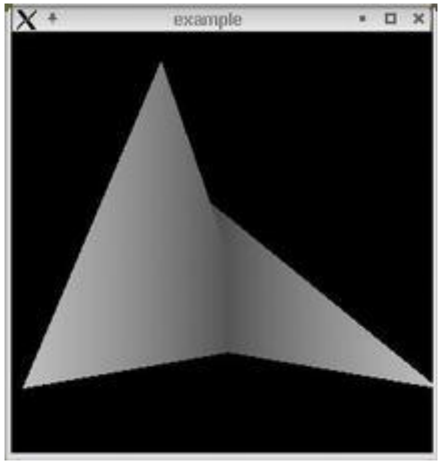
Depth buffer

Z-buffer Algorithm

- Whenever a new pixel is drawn, the Z-buffer algorithm compares the new depth information with the previous depth information in the z-buffer.
- Polygons can be drawn in any direction and can intersect.



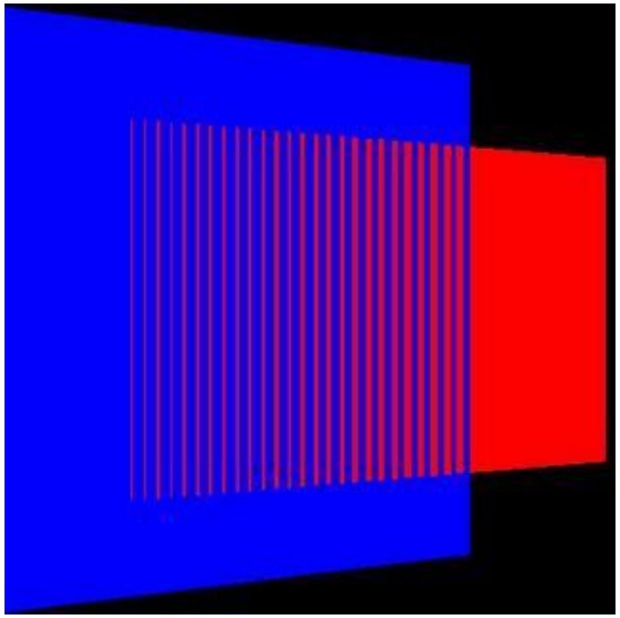
Color buffer



Depth buffer

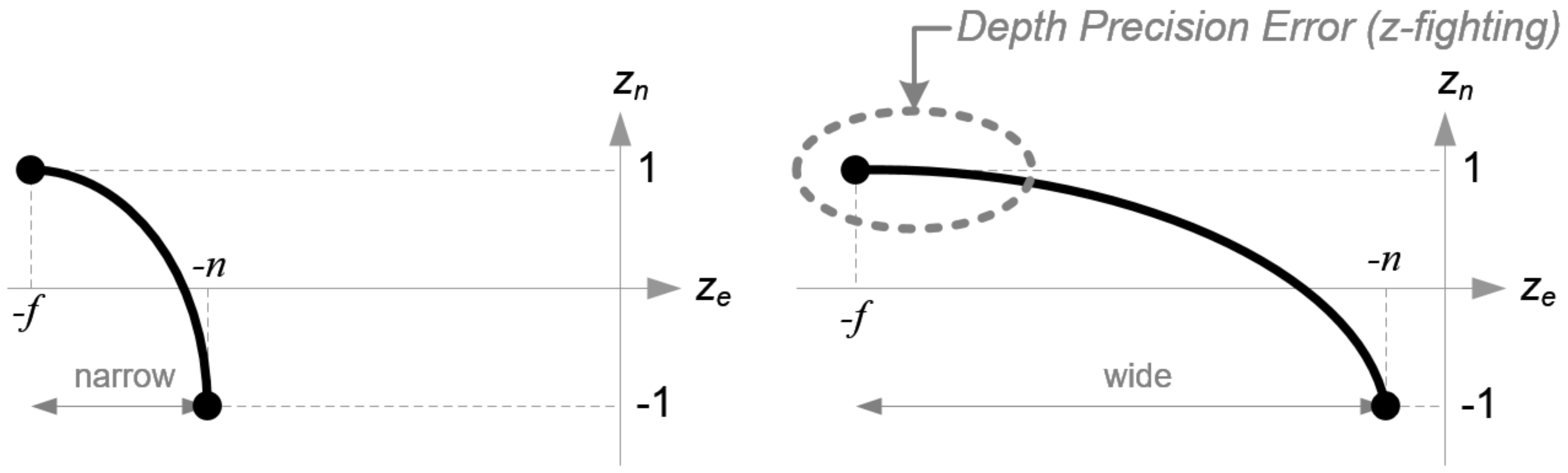
Depth Fighting

- The depth value of the Z-buffer has a limited resolution.
- The overlap of polygons with a depth value that is very close to the depth buffer creates "depth-fighting".
- This is a phenomenon that occurs due to "floating point round-off errors" when polygons are drawn, where random parts of polygons flight for rendering each other.



Depth Fighting

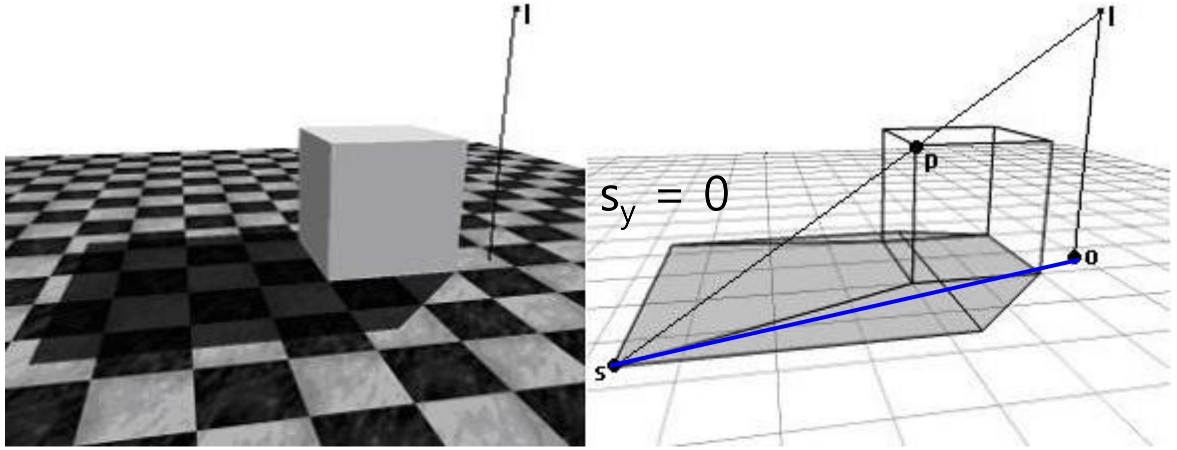
- There is very high precision at the *near* plane, but very little precision at the *far* plane.
- If the range [-n, -f] is getting larger, it causes a depth precision problem (z-fighting); a small change of z_e around the *far* plane does not affect on z_n value.
- The distance between *n* and *f* should be short as possible to minimize the depth buffer precision problem.



Depth Fighting

- Z-fighting can be **reduced through the use of a higher resolution depth buffer**, by z-buffering in some scenarios, or by **simply moving the polygons further apart**.
- Z-fighting that is caused by insufficient precision in the depth buffer can be **resolved by simply reducing the visible distance in the world**. **This reduces the distance between the near and far planes and solves the precision issue**.
- Another technique that is utilized to **reduce or completely eliminate Z-fighting is switching to a logarithmic Z-buffer, reversing Z**. Due to the way they are encoded, floating-point numbers have much more precision when closer to 0. Here, **reversing Z** leads to more precision when storing the depth of very distant objects, hence greatly reducing Z-fighting.

Planar Shadow [J. Blinn, 88]



$$\frac{l_y - p_y}{p_x - l_x} = \frac{l_y - 0}{\mathbf{s}_x - l_x}$$

$$\Rightarrow \frac{\mathbf{s}_x - l_x}{p_x - l_x} = \frac{l_y - 0}{l_y - p_y}$$

$$\mathbf{s}_x = \frac{l_y(p_x - l_x)}{l_y - p_y} + l_x$$

$$t = \frac{l_y}{l_y - p_y}$$

$$\begin{bmatrix} s_x \\ 0 \\ s_z \\ 1 \end{bmatrix} = \begin{bmatrix} l_y & -l_x & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -l_z & l_y & 0 \\ 0 & -1 & 0 & l_y \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

$$\mathbf{s} = l + t(p - l)$$

$$\mathbf{n} \cdot \mathbf{s} + d = 0$$

$$t = \frac{l_y}{l_y - p_y}$$

Planar Shadow [J. Blinn, 88]

$$\mathbf{s} = l + \frac{l_y}{l_y - p_y}(p - l)$$

$$s_x = l_x + \frac{l_y}{l_y - p_y}(p_x - l_x)$$

$$s_y = 0$$

$$s_z = l_z + \frac{l_y}{l_y - p_y}(p_z - l_z)$$

$$s_x = l_x + \frac{l_y}{l_y - p_y}(p_x - l_x) = \frac{l_x(l_y - p_y) + l_y(p_x - l_x)}{l_y - p_y} = \frac{l_y p_x - l_x p_y}{l_y - p_y}$$

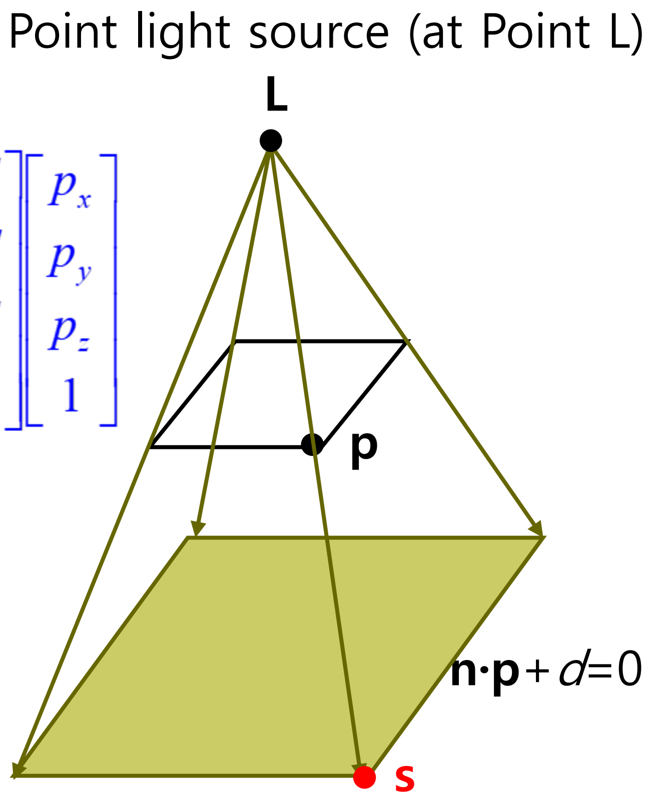
$$s_z = l_z + \frac{l_y}{l_y - p_y}(p_z - l_z) = \frac{l_z(l_y - p_y) + l_y(p_z - l_z)}{l_y - p_y} = \frac{-l_z p_y + l_y p_z}{l_y - p_y}$$

$$\begin{bmatrix} s_x \\ 0 \\ s_z \\ 1 \end{bmatrix} = \begin{bmatrix} l_y & -l_x & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -l_z & l_y & 0 \\ 0 & -1 & 0 & l_y \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

$s_x = l_y p_x - l_x p_y + 0p_z$
$s_y = 0$
$s_z = 0p_x - l_z p_y + l_y p_z$
$w = 0p_x - p_y + 0p_z + l_y$

Projection Shadow

$$\begin{bmatrix} s_x \\ s_y \\ s_z \\ 1 \end{bmatrix} = \begin{bmatrix} n\cdot l + d - l_x n_x & -l_x n_y & -l_x n_z & -l_x d \\ -l_y n_x & n\cdot l + d - l_y n_y & -l_y n_z & -l_y d \\ -l_z n_x & -l_z n_y & n\cdot l + d - l_z n_z & -l_z d \\ -n_x & -n_y & -n_z & n\cdot l \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$



$$\mathbf{s} = l + t(p - l)$$

$$\mathbf{n}\cdot\mathbf{s} + d = 0$$

$$\mathbf{n}\cdot(l + t(p-l)) + d = 0$$

$$\mathbf{n}\cdot l + t(\mathbf{n}\cdot(p-l)) = -d$$

$$t(\mathbf{n}\cdot p - \mathbf{n}\cdot l) = -d - \mathbf{n}\cdot l$$

$$t = \frac{-d - \mathbf{n}\cdot l}{\mathbf{n}\cdot p - \mathbf{n}\cdot l} \qquad \therefore\ \mathbf{s} = l + \left[\frac{\mathbf{n}\cdot l + d}{\mathbf{n}\cdot l - \mathbf{n}\cdot p}\right](\mathbf{p} - l)$$

Projection Shadow

$$\mathbf{s} = l + \frac{n\cdot l + d}{n\cdot l - n\cdot p}(p - l)$$

$$s_x = l_x + \frac{n\cdot l + d}{n\cdot l - n\cdot p}(p_x - l_x)$$

$$s_y = l_y + \frac{n\cdot l + d}{n\cdot l - n\cdot p}(p_y - l_y)$$

$$s_z = l_z + \frac{n\cdot l + d}{n\cdot l - n\cdot p}(p_z - l_z)$$

$$s_x = l_x + \frac{n\cdot l + d}{n\cdot l - n\cdot p}(p_x - l_x)$$

$$= \frac{l_x(n\cdot l - n\cdot p) + (n\cdot l + d)p_x - (n\cdot l + d)l_x}{n\cdot l - n\cdot p}$$

$$= \frac{l_x n\cdot l - l_x n_x p_x - l_x n_y p_y - l_x n_z p_z + (n\cdot l + d)p_x - l_x n\cdot l - l_x d}{-n_x p_x - n_y p_y - n_z p_z + n\cdot l}$$

$$= \frac{(n\cdot l + d - l_x n_x)p_x - l_x n_y p_y - l_x n_z p_z - l_x d}{-n_x p_x - n_y p_y - n_z p_z + n\cdot l}$$

$$\begin{bmatrix} s_x \\ s_y \\ s_z \\ 1 \end{bmatrix} = \begin{bmatrix} n\cdot l + d - l_x n_x & -l_x n_y & -l_x n_z & -l_x d \\ -l_y n_x & n\cdot l + d - l_y n_y & -l_y n_z & -l_y d \\ -l_z n_x & -l_z n_y & n\cdot l + d - l_z n_z & -l_z d \\ -n_x & -n_y & -n_z & n\cdot l \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

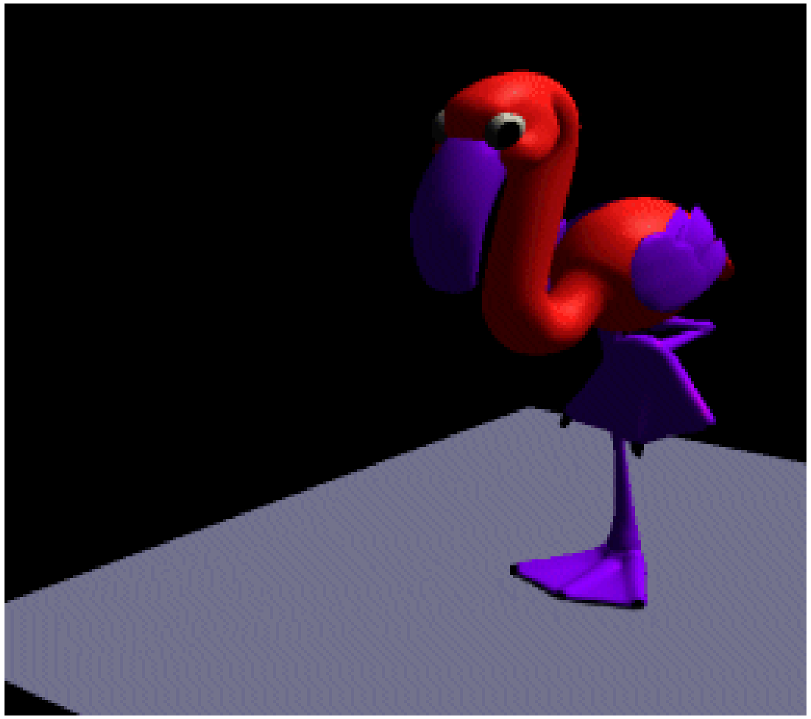
Projection Shadow Matrix

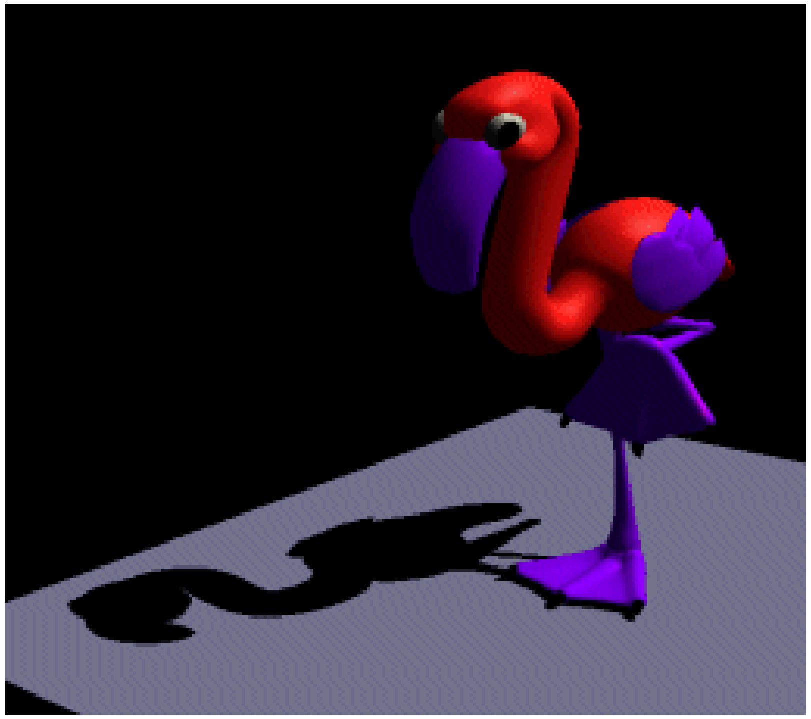
$$\begin{bmatrix} s_x \\ s_y \\ s_z \\ 1 \end{bmatrix} = \begin{bmatrix} n\cdot l - l_x n_x & -l_x n_y & -l_x n_z & -l_x n_w \\ -l_y n_x & n\cdot l - l_y n_y & -l_y n_z & -l_y n_w \\ -l_z n_x & -l_z n_y & n\cdot l - l_z n_z & -l_z n_w \\ -l_w n_x & -l_w n_y & -l_w n_z & n\cdot l - l_w n_w \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

투영평면 $n = [n_x, n_y, n_z, n_w]$

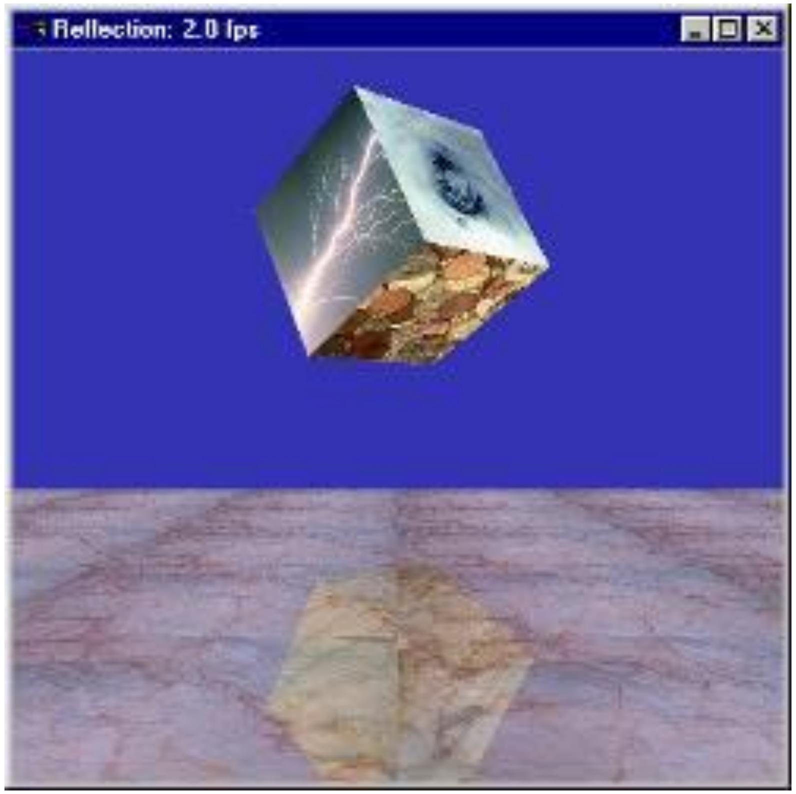
광원 $l = [l_x, l_y, l_z, l_w]$ *where* $l =$ 평행광원이면 $l_w = 0,\ l =$ 점광원이면 $l_w = 1$

Shadow


Render without shadow


Render with shadow

Reflection



http://www.gamasutra.com/features/19990723/opengl_texture_objects_02.htm

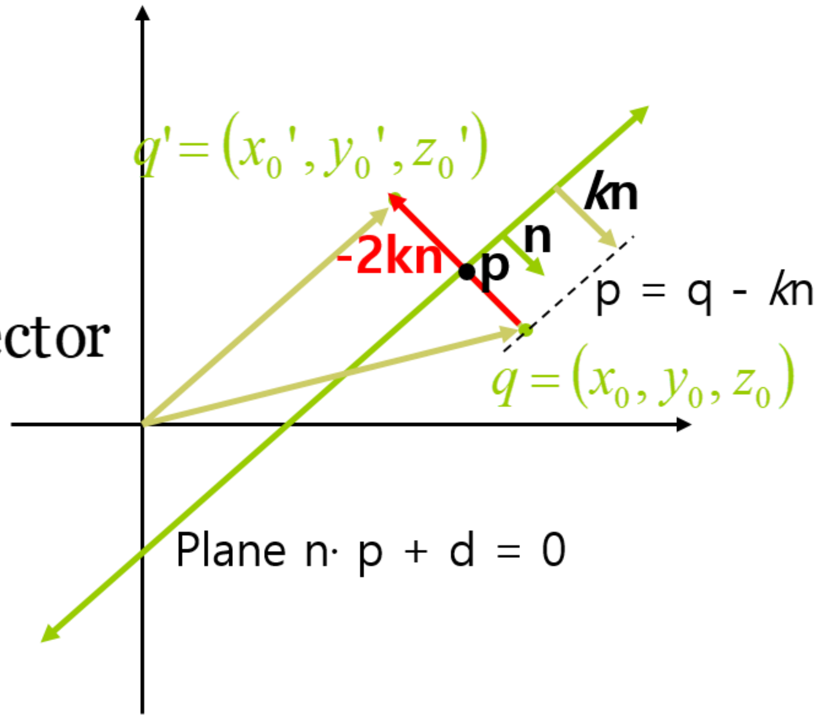
Planar Reflection

- How to calculate the reflection point, q′=(x_0′, y_0′, z_0′) of the point, q=(x_0, y_0, z_0) for the mirror plane (n, d)

$$
\begin{aligned}
q' &= q - 2k\mathrm{n} \\
&= q - 2\frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2 + b^2 + c^2}}\mathrm{n} \\
&= q - 2(n \cdot q + d)\mathrm{n} \text{ when n is unit vector}
\end{aligned}
$$

$$q' = Rq$$

$$
\mathbf{R} = \begin{bmatrix} 1-2a^2 & -2ab & -2ac & -2ad \\ -2ab & 1-2b^2 & -2bc & -2bd \\ -2ac & -2bc & 1-2c^2 & -2cd \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$



The signed shortest distance from point q to plane is k = n•q + d (if n is a unit vector)

Planar Reflection

$$\begin{bmatrix} x_0' \\ y_0' \\ z_0' \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} - 2(ax_0 + by_0 + cz_0 + d)\begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad \textit{when n is unit vector} \; (a^2 + b^2 + c^2 = 1)$$

$$\begin{aligned} x_0' &= x_0 - 2(ax_0 + by_0 + cz_0 + d)a \\ &= (1 - 2a^2)x_0 - 2aby_0 - 2acz_0 - 2ad \\ y_0' &= y_0 - 2(ax_0 + by_0 + cz_0 + d)b \\ &= -2abx_0 + (1 - 2b^2)y_0 - 2bcz_0 - 2bd \\ z_0' &= z_0 - 2(ax_0 + by_0 + cz_0 + d)c \\ &= -2acx_0 - 2bcy_0 + (1 - 2c^2)z_0 - 2cd \end{aligned}$$

$$\begin{bmatrix} x_0' \\ y_0' \\ z_0' \end{bmatrix} = R\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} 1-2a^2 & -2ab & -2ac & -2ad \\ -2ab & 1-2b^2 & -2bc & -2bd \\ -2ac & -2bc & 1-2c^2 & -2cd \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Planar Reflection

- Reflection transformation matrix for the plane (*yz-*, *xz-*, *xy-plane*)

yz-plane Plane(1,0,0,0)

$$\mathbf{R}_{yz} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

xz-plane Plane(0,1,0,0)

$$\mathbf{R}_{xz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

xy-plane Plane(0,0,1,0)

$$\mathbf{R}_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$